Optimization 2022 - Third compulsory assignment

During the lectures, you have learned about the idea of reductions - these allow you to decide instances of a problem A using an algorithm for problem B, by making an efficient translation of instances of A to instances of B.

In this compulsory assignment, you will use them as a practical tool. You will be given instances of CircuitSAT which, as you have seen in the lectures, can be reduced to SAT instances.

In this assignment, while implementing the reduction in practice, we will try to follow the conventions of the formal definition. A reduction is a map from strings to strings. Contrary to the formal definition we shall not restrict ourselves to binary strings. Instead, we shall view arbitrary text-files as strings. A reduction thus maps text files to text files. In order to model the languages CircuitSAT and SAT we shall fix encodings of Boolean circuits and CNFs, described below. The language Circuit-SAT is then the set of text-files containing valid encodings of Boolean circuits (in the Optimization circuit format) with n inputs and a single output, such that there exists an assignment to the inputs that make the circuit evaluate to 1. Analogously, the language SAT is the set of text-files containing valid encodings (in the DIMACS CNF format) of CNFs that are satisfiable.

Each circuit file (ending on .circuit) is structured as follows:

1. The first line denotes the number of input gates.
2. Then, each following line will consist of a character from $\{0,1,A,O,X,E,N,C\}$ specifying a FALSE, TRUE, AND, OR, XOR, EQUALITY, NOT or COPY-gate. The inputs to the gate are specified by a (possibly empty) list of numbers listed after the character.
3. The inputs are numbered 1 through n. The other gates are numbered implicitly by their line number, starting from n+1.
4. The output of the circuit is the last gate

For example, consider the following .circuit-file:

```
3
A 1 2
X 2 3
O 4 5
N 6
```

The circuit has three input gates and 4 other gates. It contains an AND-gate that takes the first two input gates as input, as well as a XOR gate with the second and third input gate as input. Then, an OR-gate is evaluated on the outputs of the AND- and XOR-gate, which is then negated afterwards. The negated value is the output of the circuit.

In the anaconda Python distribution there is a nice interface to the picosat SAT solver called pycosat. You can easily use it, as described in `https://pypi.org/project/pycosat/`. It accepts as inputs CNFs which are essentially in the DIMACS CNF format. A very intuitive description of the DIMACS file format can be found in `https://people.sc.fsu.edu/~jburkardt/data/cnf/cnf.html`.

This assignment consists of the following tasks, supported by the provided template code `circuitsat.py`

1. Implement the reduction from CircuitSAT to SAT (by implementing the functions `read_circuit_file`, `CSAT_to_SAT`, and `reduce_CSAT_to_SAT`. Your reduction should work even if the input file is just "garbage".

2. Consider the following variation of CircuitSAT, called CircuitSAT$_{\geq 2}$: A circuit C belongs to CircuitSAT$_{\geq 2}$ if it has at least two different satisfying assignments. Implement the reduction from CircuitSAT to SAT (by implementing the function `reduce_CSAT2_to_SAT`).

3. Test your reductions and solve the resulting CNFs with pycosat. You should make your own test cases in addition to the supplied ones.

4. Upload your completed code `circuitsat.py` together with a high level explanation of why your solutions are correct.